# Towards a Hybrid Formal Method for Swarm-Based Exploration Missions

Christopher A. Rouff
SAIC
Advanced Concepts Business Unit
McLean, VA 22102
rouffc@saic.com

Michael G. Hinchey, James L. Rash, Walter F. Truszkowski
NASA Goddard Space Flight Center
Information Systems Division
Greenbelt, Maryland, USA
{michael.g.hinchey, james.l.rash,
walter.f.truszkowski}@nasa.gov

## Abstract

*NASA is investigating the use of swarms of robotic vehicles for future space exploration missions. Such swarms offer many advantages of traditional, single spacecraft, missions. Intelligent swarms offer potential for self-management and survivability, and their emergent properties make such swarms potentially very powerful. However, they are significantly more difficult to design, and ensuring that proper behaviors will emerge is a complex task. NASA's FAST project is investigating the use of formal approaches to the specification and verification of such systems. Using ANTS, a NASA concept mission, as a case study, multiple formal methods were evaluated to determine their effectiveness in modeling and ensuring desired swarm behavior. We discuss this evaluation and propose a hybrid formal method for use in the development of future NASA intelligent swarms.*

**Key Words**: Verification, Formal Methods, Swarm Technology

## 1 Introduction

The use of swarm technologies has been identified as a means of conducting new types of science and mitigating risks in future NASA exploration missions. Traditional exploration mission concepts, involving a single large spacecraft, are being supplemented with missions that involve several smaller spacecraft, operating in collaboration, analogous to swarms in nature.

This enables NASA to send spacecraft to explore regions of space where traditional craft or manned missions would be impractical. A swarm-based approach also provides greater redundancy (and, consequently, greater protection of assets), and reduced costs and risk.

Planned missions entail the use of several unmanned autonomous vehicles (UAVs) flying approximately one meter above the surface of Mars; the use of armies of tetrahedral walkers to explore the Mars and Lunar surface; constellations of satellites flying in formation; and, the use of miniaturized pico-class spacecraft to explore the asteroid belt.

A NASA project, Formal Approaches to Swarm Technology (FAST), is investigating the requirements of appropriate formal methods for use in such missions, and is beginning to apply these techniques to specifying and verifying parts of a future NASA swarm-based mission.

We give a brief overview of swarm technologies and, in particular, ANTS, a NASA concept mission based on the use of swarm technology. We present the results of evaluation of a number of formal methods for verifying swarm-based missions, and propose a hybrid formal method for verifying swarm-based systems, which has been applied to parts of the ANTS mission.

## 2 Swarm Technologies

A swarm [2] consists of a large number of simple agents that have local interactions (interactions between agents and the environment). There is no central controller directing the swarm; they are self-organizing based on the emergent behavior of the simple interactions. This emergent behavior is sometimes referred to as the macroscopic behavior, while the individual behavior and local interactions are referred to as the microscopic behavior. These types of swarms exhibit self-organization since there is no external force directing their behavior and no single agent has a global view of the intended macroscopic behavior. This type of behavior is observed in insects and flocks of birds. Bonabeau et al. [3], who studied self-organization in social insects, state that "complex collective behaviors may emerge from interactions among individuals that exhibit simple behaviors" and describe emergent behavior as "a set of dynamical mechanisms whereby structures appear at the global level of a system from interactions among its lower-level components."

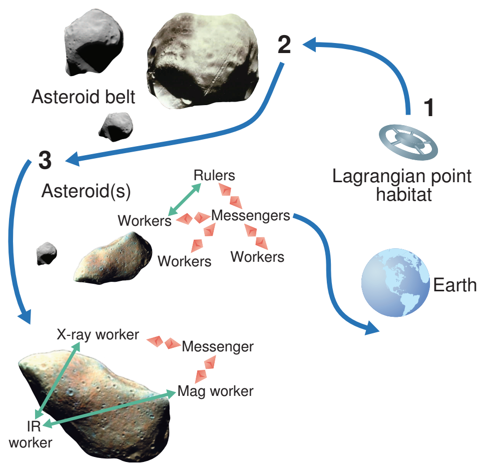

**Figure 1. ANTS Mission Concept**

Intelligent swarm technology is based on swarm technology where the individual members of the swarm also exhibit independent intelligence [1]. With intelligent swarms, members of the swarm may be heterogeneous or homogeneous. Even if the swarm starts as homogeneous, members of the swarm may learn different things due to their differing environments, consequently develop different goals, and thereby become a heterogeneous swarm. Intelligent swarms may also be made up of heterogeneous elements from the outset, reflecting different capabilities as well as a possible social structure. Verifying such systems becomes even more difficult, since the swarms are no longer made up of homogeneous members with limited intelligence and communications. Verifying intelligent swarms will be difficult, not only due to the complexity inherent in each member, but also due to the complex interaction of a large number of intelligent elements. This creates a huge state-space, and since the elements may be learning, the behavior of individual elements and emergent behavior of the swarm will be constantly changing and difficult to predict.

## 3 ANTS Mission Overview

The Autonomous Nano-Technology Swarm (ANTS) mission [6, 7] will involve the launch of a swarm of autonomous pico-class (approximately 1kg) spacecraft that will explore the asteroid belt for asteroids with certain characteristics. Figure 1 gives an overview of the ANTS mission [17]. In this mission, a transport ship, launched from Earth, will travel to a point in space (a Lagrangian point) where net gravitational forces on small objects (such as spacecraft) are negligible. From this point, 1000 spacecraft, that have been assembled en route from Earth, will be launched into the asteroid belt. The tiny spacecraft, having no on-board propulsion, and only solar sails to provide thrust, will have severe limits on their ability to maneuver during operations. Consequently, collisions with asteroidal bodies and with each other will be likely, so that over the life of the exploration mission, 60 to 70 percent of the swarm is expected to be lost.

Because of their small size, each spacecraft will carry just one specialized instrument for collecting a specific type of data from asteroids in the belt. As a result, the ANTS spacecraft will cooperate and coordinate using a hierarchical social behavior analogous to colonies or swarms of insects, with some spacecraft directing others. Approximately 80 percent of the spacecraft will be *workers* that will carry the specialized instruments (e.g., a magnetometer, x-ray, gamma-ray, visible/IR, neutral mass spectrometer) and will obtain specific types of data. Some will be coordinators (called *leaders* or *rulers*) that have rules that determine the types of asteroids and data that the mission is interested in, and that will coordinate the efforts of the workers. The third type of spacecraft are *messengers* that will coordinate communication between the rulers and workers, and communications with the mission control center on Earth.

The swarm will form sub-swarms under the control of a ruler, which contains models of the types of science that are to be pursued. The ruler will coordinate workers each of which uses its individual instrument to collect data on specific asteroids. This information is fed back to the ruler, who determines which asteroids are worth examining further. If the data matches the profile of an asteroid that is of interest, an imaging spacecraft will be sent to the asteroid to ascertain the exact location and to create a rough model to be used by other spacecraft for maneuvering around the asteroid. Other teams of spacecraft will then coordinate to finish mapping the asteroid to form a complete model.

## 4 Specifying and Verifying ANTS

The above is a very simplified description of the ANTS mission. For a more detailed exposition, the interested reader is directed to [13, 17], or to the ANTS website. But, as can be seen from the brief exposition above, ANTS is a highly complex system that poses many significant challenges. Not least amongst these are the complex interactions between heterogeneous components, the need for continuous re-planning, re-configuration and re-optimization, the need for autonomous operation without intervention from Earth, and the need for assurance of the correct operation of the mission [13].

Increasing mission software complexity increases the difficulty of finding errors and fully testing the system. Many behaviors, including those that produce race conditions, for example, are time-based and only occur when processes send or receive data at particular times or in a particular sequence, or after learning occurs. Such error conditions can rarely be found by inputting sample data and checking whether the results are correct. To find these errors through testing, the software processes involved would have to be executed in all possible combinations of states (state space) that the processes could collectively be in. Because the state space is exponential (and sometimes factorial) to the number of states, it becomes untestable with a relatively small number of processes. Traditionally, to get around the state explosion problem, testers have artificially reduced the number of states of the system and approximated the underlying software using models. This reduces the fidelity of the model and can mask potential errors.

## 5 Formal Methods

Formal methods are proven approaches for assuring the correct operation of complex interacting systems [9]. Formal methods are mathematically-based tools and techniques for specifying and verifying systems. They are particularly useful for specifying complex parallel and distributed systems where the entire system is difficult for a single person to fully understand and when more than one person was involved in the development. Once written, a formal specification can be used to prove properties of a system correct (e.g., the underlying system will go from one state to another or not into a specific state), used to check for particular types of errors (e.g., race conditions), or used as input to a model checker.

Verifying emergent behavior is one area that most formal methods have not addressed. However, formal methods can provide guidance in determining possible emergent behaviors that must be considered. Formal methods have been widely used for test case generation to develop effective test cases whereby the vast majority of the critical code is tested well. (This differs from haphazard testing techniques that may waste time testing code that will never execute.) Similar techniques may be used with formal methods, not to generate a test plan, but to propose certain properties that might or might not hold, or certain emergent behaviors that might arise.

With formal methods we may propose that certain properties hold, and prove that they hold. In particular this is invaluable for properties that we cannot test on Earth, or can test only with very expensive simulation. By its nature, a good formal specification can guide us to propose and verify certain behaviors (or lack of certain behaviors) that we would often not think of when using regular testing techniques. Moreover, if properly applied, and properly used in the development process, a good formal specification can help us to prove the presence or absence of particular properties in the overall system well in advance of mission launch, or even implementation. Indeed, various formal methods offer support for simulation and automatic code generation, making the initial investment well worth while.

## 6 Formal Methods for Intelligent Swarms

The FAST (Formal Approaches to Swarm Technologies) project has surveyed various formal methods and various formal techniques to determine whether an existing formal method, or a combination of existing methods, could be suitable for specifying and verifying swarm based missions such as ANTS, and their emergent behavior. Various methods were surveyed based on a small number of criteria that were determined to be important in their application to intelligent swarms. The results of this survey can be found in [12]. Although the survey identified a few formal methods that have been used to specify swarm-based systems, initially only two formal approaches were found that had been used to analyze the emergent behavior of swarms, namely Weighted Synchronous Calculus of Communicating Systems (WSCCS) and Artificial Physics [14]. Since the survey was completed, two other approaches that may prove valuable in analyzing emergent behavior—CommUnity [8] and CSP2B [4]—have been brought to our attention, although we have not as yet identified their use with swarm technologies *per se*.

## 7 Specifications of the Virtual Experiment

A virtual experiment is conducted in the ANTS mission by a subset consisting of a Leader spacecraft and individual worker spacecraft. Details of the operations of the ANTS mission can be found on the ANTS web page. A scenario for the ANTS mission is based on the ANTS targeting an asteroid on which to conduct an experiment, and then forming a team to carry out that experiment. The following is a brief description of the scenario:

Team leaders contain models of the types of science they wish to perform. Parts of this model are communicated to the messenger spacecraft, which then relay it on to the worker spacecraft. The worker spacecraft take measurements of asteroids using whichever kind of instrument they have, until something is found that matches the goal that was established by the leader.

The data will then be sent to a messenger to be conveyed back to the leader. If the data matches the profile of the type of asteroid that is being sought, an imaging spacecraft will be sent to the asteroid to ascertain the exact position and orbital parameters and to create a rough model prior to the arrival of other spacecraft, so that they will be supplied with a model to use for maneuvering around the asteroid.

Other spacecraft that would then work together to finish the model and mapping of the asteroid would include:

- an asteroid detector/stereo mapper team that would consist of two spacecraft with field-imaging spectrometers, and a dynamic modeler with an enhanced radio science instrument for measuring dynamic properties (such as spin, density, and mass distribution);

- a petrologist team that would consist of X-ray, Near Infrared, Gamma-ray, Thermal IR and wide-field imager to determine the distribution of elements, minerals and rocks present;

- a photogeologist team consists of Narrow Field and Wide Field Imagers and Altimeter to determine the nature and distribution of geological units based on texture, albedo, color, and apparent stratigraphy;

- a prospector team consisting of an altimeter, magnetometer, near infrared, infrared, and X-ray spectrometers to determine the distribution of resources.

The above teams would work together to form a model of asteroids as well as to form virtual instruments.

Many things can happen when an ANTS team encounters an asteroid. A spacecraft can perform a flyby and make opportunistic observations. The flyby can be used to first determine whether the asteroid is of interest before sending an entire team to the asteroid, or to determine that, due to the nature of the instrument on the spacecraft, only a flyby is necessary. If the asteroid is of interest, a mapping spacecraft will map the asteroid and determine its size, rate and axis of rotation, whether the asteroids have any satellites/moons, etc. This information is passed on to other spacecraft that will be performing observations and need to conduct a flyby, enter an orbit around the asteroid, etc. As more data is obtained about the asteroid, other ANTS may be sent to the asteroid for further data gathering.

The following gives partial specifications of the ANTS mission using CSP, WSCCS, Unity Logic, and X-Machines. Due to space requirements only samples of the specifications are given.

## 7.1 CSP specification of ANTS

The following is a specification of the behavior of the NASA ANTS mission using Communicating Sequential Processes (CSP) [10]. In the specification, each of the spacecraft has goals to fulfill its mission. The aggregate or emergent behavior of all these goals should equal the goals of the mission. The following is the top-level specification of the ANTS mission:

$$\begin{aligned} ANTS_{goals} = {} & Leader_{i,l\_goals} \parallel Messenger_{j,m\_goals} \parallel \\ & Worker_{ki,w\_goals} \\ & \bullet 1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq p \end{aligned}$$

where *m* is the number of leader spacecraft, *n* the number of messenger spacecraft and *p* the number of worker spacecraft. The ANTS mission is initialized with a set of goals given to it by the principal investigator, and part of these goals are given to the leader (some may not be given to the leader because they are ground based or not applicable to the leader). Each spacecraft is also given a name (in this case a number) so it can identify itself when communicating with the other spacecraft and with Earth.

The leader specification consists of two processes, the communications process and the intelligence process:

$$\begin{aligned} Leader_i = {} & LEADER\_COM_{i,\{\}} \parallel \\ & LEADER\_INTELLIGENCE_{i,goals,model} \end{aligned}$$

The communication process, LEADER_COM, specifies the behavior of the spacecraft as it relates to communicating with the other spacecraft. The second process, LEADER_INTELLIGENCE, is the specification of the intelligence of the leader. This is where the deliberative and reactive parts of the intelligence are implemented and the maintenance of the goals for the leader is undertaken. In addition to the goals, the LEADER_INTELLIGENCE process also maintains the models of the spacecraft and its environment and specifies how it is modified during operations. Each of the above processes has parameters that have an identifying number to identify a spacecraft within a group, as well as other parameters that are sets that store conversations, goals and models. Since at startup there have been no conversations, the conversation set in the LEADER_COM process is empty. Since leaders are given initial goals and models, these sets are non-empty at start up. The following is the top-level specification of the leader communication.

$$\begin{aligned} & LEADER\_COM_{i,conv} = leader.in?msg \rightarrow \\ & \quad \text{case } LEADER\_MESSAGE_{i,conv,msg} \\ & \qquad\quad \text{if } sender(msg) = LEADER \\ & \qquad MESSENGER\_MESSAGE_{i,conv,msg} \\ & \qquad\quad \text{if } sender(msg) = MESSENGER \\ & \qquad WORKER\_MESSAGE_{i,conv,msg} \\ & \qquad\quad \text{if } sender(msg) = WORKER \\ & \qquad EARTH\_MESSAGE_{i,conv,msg} \\ & \qquad\quad \text{if } sender(msg) = EARTH \\ & \qquad ERROR\_MESSAGE_{i,conv,msg} \\ & \qquad\quad \text{otherwise} \end{aligned}$$

The above shows the messages from other spacecraft types that a leader may receive. Messages sent from another leader may be one of two types: requests or informational. For requests, the requests may be for such things as information on the leader's model or goals, for resources (e.g., more workers), or for status. Messages may also be informational and contain data containing new goals or new information for the agent's model (e.g., due to a new discovery). This information needs to be examined by the intelligence process and the model process to determine whether any updates to the goals or model are required. The following processes further describe the messages that may be received from other leaders.

$$
\begin{aligned}
&LEADER\_MESSAGE_{i,conv} = \\
&\quad \text{case } LEADER\_INFORMATION_{i,conv,msg} \\
&\qquad \text{if } content(msg) = information \\
&\quad\; LEADER\_REQUESTS_{i,conv,msg} \\
&\qquad \text{if } content(msg) = request \\
&\quad\; LEADER\_RECEIVE_{i,conv,msg} \\
&\qquad \text{if } content(msg) = reply\_to\_request \\
&\quad\; ERROR\_MESSAGE_{i,conv,msg} \\
&\qquad \text{otherwise}
\end{aligned}
$$

The following gives additional information on the leader information messages.

$$
\begin{aligned}
&LEADER\_INFORMATION_{i,conv} = \\
&\quad leader\_model_i(NEW\_INFO, msg) \\
&\quad \rightarrow goals\_channel_i(NEW\_INFO, msg) \\
&\quad \rightarrow LEADER\_COM_{i,conv}
\end{aligned}
$$

If the message is new information, then that information has to be sent to the deliberative part of the agent to check whether the goals should be updated as well as the model part to check whether any of the information requires updates to the model.

## 7.2 WSCCS Specification of ANTS

To model the ANTS Leader spacecraft, WSCCS [16], a process algebra, takes into account:

- The possible states of the Leader;
- Actions taken in each agent-state that would qualify them to be "in" those states;
- The relative frequency of each action for the agent;
- The priority of each action for that agent.

**Table 1. Agent state and actions**

| Agent State | Actions leading to the agent state | $f$ | $p$ |
|---|---|---|---|
| | Identity | | |
| Communicating | SendMessageWorker | 50 | 2 |
| | SendMessageLeader | 50 | 2 |
| | SendMessageError | 1 | 1 |
| | ReceiveMessageWorker | 50 | 2 |
| | ReceiveMessageLeader | 50 | 2 |
| | ReceiveMessageError | 1 | 1 |
| Reasoning | ReasoningDeliberatve | 50 | 2 |
| | ReasoningReactive | 50 | 2 |
| Processing | ProcessingSortingAndStorage | 17 | 2 |
| | ProcessingGeneration | 17 | 2 |
| | ProcessingPrediction | 17 | 2 |
| | ProcessingDiagnosis | 16 | 2 |
| | ProcessingRecovery | 16 | 2 |
| | ProcessingRemediation | 17 | 2 |

Consider the following actions, agent states, and view of frequency, *f*, and priority, *p*, of the actions of the Leader as seen in Table 1. Based on this information, WSCCS provides an algebra by which the behavior of the Leader can be studied and verified. Given the information from the table above, we define the agent-states as:

$$
\begin{aligned}
&Communicating \equiv \\
&50\omega^2 : SendMessageWorker.Communicating \\
&+50\omega^2 : SendMessageLeader.Communicating \\
&+1\omega^1 : SendMessageError.Communicating \\
&+50\omega^2 : ReceiveMessageWorker.Communicating \\
&+50\omega^2 : ReceiveMessageLeader.Communicating \\
&+1\omega^1 : ReceiveMessageError.Communicating \\
&+50\omega^2 : ReasoningDeiberative.Reasoning \\
&+50\omega^2 : ReasoningReactive.Reasoning \\
&+17\omega^2 : ProcessingSortingAndStorage.Processing \\
&+17\omega^2 : ProcessingGeneration.Processing \\
&+17\omega^2 : ProcessingPrediction.Processing \\
&+16\omega^2 : ProcessingDiagnosis.Processing \\
&+16\omega^2 : ProcessingRecovery.Processing \\
&+17\omega^2 : ProcessingRemediation.Processing
\end{aligned}
$$

The symbol $+$ in this notation denotes that the Communicating Leader will make a choice between the various allowed actions, and that that choice will be made based on the frequencies and priorities of each allowable action. For example, the Communicating leader may choose to remain in the Communicating state by choosing to send a message to a worker. It would do so with a frequency of 50 and a priority of 2 which tells us that it will make this choice with a probability of 12.5%. The Communicating Leader may instead choose to transition to a Processing state by processing for Recovery. There is a 4% chance that the Leader will make this choice. What follows are similar statements for the Reasoning Leader and the Processing Leader:

$Reasoning \equiv$
$50\omega^3 : ReasoningDeiberative.Reasoning$
$+50\omega^3 : ReasoningReactive.Reasoning$
$+50\omega^2 : SendMessageWorker.Communicating$
$+50\omega^2 : SendMessageLeader.Communicating$
$+1\omega^1 : SendMessageError.Communicating$
$+50\omega^2 : ReceiveMessageWorker.Communicating$
$+50\omega^2 : ReceiveMessageLeader.Communicating$
$+1\omega^1 : ReceiveMessageError.Communicating$
$+17\omega^2 : ProcessingSortingAndStorage.Processing$
$+17\omega^2 : ProcessingGeneration.Processing$
$+17\omega^2 : ProcessingPrediction.Processing$
$+16\omega^2 : ProcessingDiagnosis.Processing$
$+16\omega^2 : ProcessingRecovery.Processing$
$+17\omega^2 : ProcessingRemediation.Processing$

In the above definition of the Reasoning Leader, we see that the Leader will not choose to send or receive a message in error since the priorities of these actions are lower than the priorities of other actions.

$Processing \equiv$
$17\omega^2 : ProcessingSortingAndStorage.Processing$
$+17\omega^2 : ProcessingGeneration.Processing$
$+17\omega^2 : ProcessingPrediction.Processing$
$+16\omega^2 : ProcessingDiagnosis.Processing$
$+16\omega^2 : ProcessingRecovery.Processing$
$+17\omega^2 : ProcessingRemediation.Processing$
$+50\omega^3 : ReasoningDeiberative.Reasoning$
$+50\omega^3 : ReasoningReactive.Reasoning$

This statement shows that the Processing Leader is forced to go into the Reasoning state prior to entering the Communication State to ensure that the Leader has reasoned about its mission goals and model after processing, and before communicating to other members of the swarm.

The operations of choice $(+)$ and composition of actions $(*)$ are then defined by the following rules:

$$n\omega^{k+l} + m\omega^k = n\omega^{k+l} = m\omega^k + n\omega^{k+l}$$
$$n\omega^k + m\omega^k = (n+m)\omega^k = m\omega^k + n\omega^k$$
$$n\omega^{k+l} * m\omega^k = (nm)\omega^{k+(k+l)} = m\omega^k * n\omega^{k+l}$$
$$n\omega^k * m\omega^k = (nm)\omega^{k+k} = m\omega^k * n\omega^k$$

A transitional semantics defines what series of actions are valid for a given agent, and allows us to interpret agents as finite state automata represented by a transition graph. A transition graph derived from these transitions for the ANTS Leader Spacecraft can be developed where nodes represent the agents and edges represent the weights and actions.

**Emergent Behavior of Leaders Using Probability.** Given a swarm of $n$ Leader Spacecraft, the $n$-leader swarm will tick forward in time by performing simultaneous actions one action per leader per time-step. Thus the $n$-leader swarm will perform a composition of $n$ actions, denoted with weight $m_1\omega^{k_1} * m_2\omega^{k_2} * \cdots * m_n\omega^{k_n}$, at each time-step. When this happens, the $n$-leader swarm must still behave according to the rules for composition seen earlier. This gives the $n$-leader swarm its own set of relative frequencies and priorities. Since there are $n$ Leaders and each has three states and 14 possible actions, the swarm of $n$ leaders has $3^n$ possible state sets and $14^n$ possible action compositions. There are only two possible priority values and four possible relative frequency values available and thus we can narrow down that each priority $k_i$ must be either 1 or 2 with each relative frequency $m_i$ either 1 (if the priority is 1) or 16, 17, or 50 (if the priority is 2).

Any composition that includes any leader communicating in error will have a priority less than the priority of not sending any messages in error and thus the swarm will not choose to send or receive a message in error. Thus, the remaining options for leaders in the swarm will include communicating (not in error), reasoning, and processing (either by prediction or recovery, or otherwise). Let $N_{comm}$ be the number of leaders in the swarm who choose to communicate (not in error) on a given time step. Let $N_{reason}$ be the number of leaders in the swarm who choose to reason on that time step. Let $N_{process16}$ be the number of leaders in the swarm who choose to process (by prediction or recovery) on that time step. Lastly, let $N_{process17}$ be the number of leaders in the swarm who choose to process (by other means) on that time step.

Then, each action by each leader will have priority 2 and relative frequency 16, 17 or 50. Thus, the composition of their actions will have weight

$$m_1\omega^{k_1} * m_2\omega^{k_2} * \cdots * m_n\omega^{k_n} =$$
$$(50^{N_{comm}+N_{reason}})(16^{N_{process16}})(17^{N_{process17}})\omega^{2n}$$

From this weighting, we can see that drastically higher frequencies exist when a larger number of leaders in the swarm choose to communicate or reason. Much lower frequencies exist when larger numbers of leaders choose to process. Thus, the swarm will be communicating and reasoning much more often than processing, although processing will take place.

**Emergent Behavior of a Leader Using Markov Chains** Markov Chains gives a different view of the Leader's emergent behavior. Based on the above statements and the previous frequencies and priorities, we can calculate the probabilities of the Leader choosing each action and therefore the probabilities that the Leader will transition to one state or another. From these probabilities we can construct the following matrix, $P$, which for each entry $p_{ij}$ shows the probability of the Leader choosing to transition from state i to state j. For example, $p_{13} = 0.25$ means that the probability of transitioning from state 1 (Initial state or Identity State) to state 3 (Processing) is 25%.

$$P = \begin{pmatrix} 0 & .5 & .25 & .25 \\ 0 & .5 & .25 & .25 \\ 0 & .5 & .25 & .25 \\ 0 & 0 & .5 & .5 \end{pmatrix}$$

Given this matrix, we can calculate the various powers, $P^n$, of the matrix. The $n$th power of the matrix $P$ will tell us the probabilities that the Leader will be in on the $n$th time step. For example, consider the results showing the calculated matrix $P^2$ in Figure 2.

We see in the matrix for $P^2$ that the entry $P^2_{242}$ is 0.25. This tells us that if the Leader begins in the fourth state (Processing), it has a probability of 25% of being in the second state (Communicating) on the second time-step. Observe the convergence of these matrices at higher powers (i.e., as time progresses) in Figures 2 and 3.

We see the powers of $P$ converging to the matrix

$$P^n = \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 0 & 1/3 & 1/3 & 1/3 \\ 0 & 1/3 & 1/3 & 1/3 \\ 0 & 1/3 & 1/3 & 1/3 \end{pmatrix}$$

where the Leader will not return to the initial state but will have equal probability of being in any of the three other states given a starting point of any of the four states. This is just an example of the type of prediction that Markov Chains may be able to deliver. These concepts are currently being studied further.

## 7.3 Unity Logic Specification of ANTS

To model the ANTS Leader spacecraft with Unity Logic [5], we consider states of the Leader just as in other state-machine based specification languages, and as in WSCCS. In Unity Logic, we will consider the states of the Leader and the actions taken to make the Leader be in those states, but the notation will appear much closer to that of classical logic. Predicates are defined to represent the actions that would put the Leader into its various states. Those predicates then become statements that, if true, would mean that the Leader had performed an action that put itself into the corresponding state. This allows us to formally specify the Leader using assertions. Unity Logic then provides a logical syntax equivalent to Propositional Logic for reasoning about these predicates and the states they imply as well as for defining specific mathematical, statistical and other simple calculations to be performed.

## 7.4 X-Machines Specification of ANTS

To model the ANTS Leader spacecraft as an X-Machine [11], we must define it as a tuple

$$L = \{Input, Memory, Output, Q, \Phi, F, start, m_0\}$$

where the components of the tuple are defined as:

$$\begin{aligned} Input = \{&worker, messenger, leader, error, \\ &Deliberative, Reactive, SortAndStore, \\ &Generate, Predict, Diagnose, \\ &Recover, Remediate\} \end{aligned}$$

is a set of data. $Memory$ will be written as a tuple $m = (Goals, Model)$, where $Goals$ describes the goals of the mission and $Model$ describes the model of the universe maintained by the Leader. The initial memory will be denoted by $(Goals_0, Model_0)$. When the goals and/or model changes, the new tuple will be denoted as

$$m' = (Goals', Model')$$

$$\begin{aligned} Output = \{&SentMessageWorker, \\ &SentMessageMessenger, \\ &SentMessageLeader, SentMessageError, \\ &ReceivedMessageWorker, \\ &ReceivedMessageMessenger, \\ &ReceivedMessageLeader, \\ &ReceivedMessageError, \\ &ReasonedDeliberatively, ReasonedReactively, \\ &ProcessedSortingAndStoring, \\ &ProcessedGeneration, ProcessedPrediction, \\ &ProcessedDiagnosis, \\ &ProcessedRecovery, ProcessedRemediation\} \end{aligned}$$

is another set of data.

$$\begin{aligned} Q = \{&Start, Communicating, Reasoning, \\ &Processing\} \end{aligned}$$

is a set of states.

$$\begin{aligned} \Phi = \{&SendMessage, ReceiveMessage, Reason, \\ &Process\} \end{aligned}$$

is a set of (partial) transition functions where each transition function maps $Memory \times Input \rightarrow Output \times Memory$ as in the following:

$$\Phi(m, Worker) = (m', SentMessageWorker)$$
$$\Phi(m, Generate) = (m', ProcessGeneration)$$

$$P^2 = \begin{pmatrix} 0 & .375000000000000000 & .312500000000000000 & .312500000000000000 \\ 0 & .375000000000000000 & .312500000000000000 & .312500000000000000 \\ 0 & .375000000000000000 & .312500000000000000 & .312500000000000000 \\ 0 & .250000000000000000 & .375000000000000000 & .375000000000000000 \end{pmatrix}$$

**Figure 2. Calculated matrix $P^2$**

$$P^{10000000000} = \begin{pmatrix} 0 & .333333333333333370 & .333333333333333370 & .333333333333333370 \\ 0 & .333333333333333370 & .333333333333333370 & .333333333333333370 \\ 0 & .333333333333333370 & .333333333333333370 & .333333333333333370 \\ 0 & .333333333333333370 & .333333333333333370 & .333333333333333370 \end{pmatrix}$$

**Figure 3. Calculated matrix $P^{10000000000}$**

Then $F : Q \times \Phi \rightarrow Q$ is a next-state partial function.

X-Machines provide a highly executable environment for specifying the ANTS spacecraft. They allow for a memory to be kept and they allow for transitions between states to be seen as functions involving inputs and outputs. This allows us to track the actions of the ANTS spacecraft as well as write to memory any aspect of the goals and model. This ability makes X-Machines highly effective for tracking and effecting changes in the goals and model. However, X-Machines do not provide any robust means for reasoning about, or predicting behaviors of, one or more spacecraft, beyond standard propositional logic. This will make specifying emergent behavior difficult.

## 7.5 An Appraisal of Approaches

Based on these properties, the experiences of creating partial specifications for the ANTS Leader Spacecraft, and the needs of the ANTS mission, we draw the following conclusions about the properties needed for effective specification and emergent behavior prediction of the ANTS mission.

An effective formal method must be able to predict the emergent behavior of 1000 agents as a swarm, as well as the behavior of the individual agent. Crucial to the mission will be the ability to modify operations autonomously to reflect the changing nature of the mission and the distance and low bandwidth communications back to Earth. For this, the formal specification will need to be able to track the goals of the mission as they change, and to modify the model of the universe as new data comes in. The formal specification will also need to allow for specification of the decision making process to aid in the determination of which instruments will be needed, at what location, with what goals, etc.

Once written, the formal specification to be developed must be able to be used to prove properties of the system correct (e.g., the underlying system will go from one state to another or not into a specific state), check for particular types of errors (e.g. race conditions), as well as be used as input to a model checker.

From this we can see that the formal method must be able to track the models of the leaders and it must allow for decisions to be made as to when the data collected has met the goals. The ANTS mission details are still being determined and are changing as more research is performed. Therefore, the formal method must be flexible enough to allow for efficient changes and re-prediction of emergent behavior.

Bearing all of this in mind, the following list summarizes the properties necessary for effective specification and emergent behavior prediction of the ANTS swarm and other swarms, and looks to the existing formal methods to provide some of the desired properties.

**Processes (X-Machines, CSP)** — Processes can be specified using the various manifestations of transition functions. This property could also be more robust.

**Reasoning (Unity Logic)** — Unity Logic provides only limited capability in this area. Other forms of possibly non-standard logics may need to be employed here to allow for intelligent reasoning with uncertain and possibly conflicting information.

**How agent chooses action alternatives (WSCCS)** — A modified version of this ability from WSCCS may be used to supply an algebra for choosing between possible actions.

**Asynchronous messaging (CSP Variant)** — Messaging may not be synchronized upon or after implementation. There are variants of CSP that support asynchronous messaging.

**Message buffering (CSP Variant)** — Message buffering may be needed due to the possibly asynchronous nature of messaging between members of the swarm. There are variants of CSP that support buffering.

**Concurrent agent states for each spacecraft (WSCCS)** — This ability is solidly in place and will require only an augmentation of the notation.

**Communication protocols between agents (CSP)** — CSP allows for this as it stands.

**Adaptability to programming (X-Machines, Unity Logic)** — Any formal specification languages that are developed will need to keep in mind the ease of converting the formal specification to programs and model checkers.

**Determining whether goals have been met (None)** — The goals of each spacecraft are constantly under review. We will need to be able to specify a method by which the spacecraft will know when the goals have been met. A modification to X-Machines may be able to solve this since the goals could be tracked using X-Machines.

**Method for determining new goals (None)** — Once goals are met, new goals must be formed. We need to be able to specify a method for forming these goals. Again, a modification to X-Machines may be the best approach.

**Model checking (CSP)** — Model checking will help avoid semantic inconsistencies in the specifications.

**Tracking Models (X-Machines)** — X-Machines have the ability to track the universe model in memory but need a more robust way to detail what the model is, how it is created, and how it is modified.

**Associating agent actions with priorities (WSCCS)** — This ability is firmly in place.

**Associating agent actions with frequencies (WSCCS)** — This ability is firmly in place.

**Predicting emergent behavior (WSCCS)** — Current WSCCS abilities are not robust enough for the purpose of predicting individual and swarm emergent behavior and will need to be enhanced by greater use of Probability, Markov Chains, and/or Chaos Theory.

A blending of the above methods seems to be the best approach for specifying swarm-based systems. Blending the memory and transition function aspects of X-Machines with the priority and probability aspects of WSCCS, and other methods, may produce a specification method that will allow all the necessary aspects for specifying emergent behavior in the ANTS mission, and other swarm-based systems. The merging of these methods is currently being performed.

## 8 Future Work

Currently the FAST project is working on integrating the above formal methods to create a hybrid method suitable for specifying and verifying swarm-based systems. After integrating the formal methods, an in-depth specification of the ANTS mission, and possibly a second NASA swarm-based mission, will be developed to give examples of the use of the new formal method as well as to determine whether there are any modifications that need to be made to the new method.

In addition to the integration of the above formal methods, tools to support the new hybrid formal method are being considered. Existing tools (both commercial and public-domain), that can be modified or enhanced, are being considered. In addition, translators are being considered, that will allow us to translate from the new formal method to notations suitable for use with existing tools. Examples of some of the tools that are being examined include editors, syntax checkers, theorem provers and model checkers.

## 9 Conclusion

Future NASA missions will increasingly exploit intelligent swarm technologies in systems to conduct new science and perform unmanned exploration. These new missions will be highly autonomous and out of touch with NASA ground stations for extended periods of time due to physical restrictions on communications. In addition, such swarms may be designed with, or unintentionally exhibit, complex emergent behavior. As such, these future missions must be built with even higher levels of assurance than heretofore has been the norm.

Formal verification of swarm-based missions requires the development of an effective formal method that has the expressive power to represent large swarms, and to capture their (intentional or unintentional) emergent behavior. An appropriate method must enable us to track the goals of the mission as they change and allow for modification of the model as new data comes in. It must allow for specification of the decision-making process to aid in the decision as to which instruments will be needed, at what location, with what goals, etc.

We have identified several important attributes of a formal approach to the specification and verification of swarm-based systems. We have also surveyed a wide variety of potential formal approaches, and have identified a shortlist of several formal methods that have been used for modeling swarms or have the appropriate attributes. From this potential list we have developed sample formal specifications of parts of the NASA ANTS mission. As part of the FAST project, we are currently integrating four of these methods to develop a new hybrid formal method for swarm-based systems, which will be applied to the ANTS mission.

## Acknowledgement

This work was supported by the NASA Office of Systems and Mission Assurance (OSMA) through its Software Assurance Research Program (SARP) project, Formal Approaches to Swarm Technologies (FAST), administered by the NASA Independent Verification and Validation (IV&V) Facility.

## References

[1] G. Beni and J. Want. Swarm intelligence. In *Proc. Seventh Annual Meeting of the Robotics Society of Japan*, pages 425–428, Tokyo, Japan, 1989. RSJ Press.

[2] E. Bonabeau, M. Dorigo, and G. Théraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, 1999.

[3] E. Bonabeau and G. Théraulaz. Swarm smarts. *Scientific American*, pages 72–79, March 2000.

[4] M. J. Butler. *csp2B : A Practical Approach To Combining CSP and B*. Declarative Systems and Software Engineering Group, Department of Electronics and Computer Science, University of Southampton, February 1999.

[5] K. M. Chandy and J. Misra. *Parallel Program Design: A Foundation*. Addison-Wesley Publishing Company, 1988.

[6] P. E. Clark, S. A. Curtis, and M. L. Rilee. ANTS: Applying a new paradigm to lunar and planetary exploration. In *Proc. Solar System Remote Sensing Symposium*, Pittsburgh, Pennsylvania, USA, 20–21 September 2002.

[7] S. A. Curtis, J. Mica, J. Nuth, G. Marr, M. L. Rilee, and M. K. Bhat. ANTS (Autonomous Nano-Technology Swarm): An artificial intelligence approach to asteroid belt resource exploration. In *Proc. Int'l Astronautical Federation, 51st Congress*, October 2000.

[8] J. L. Fiadeiro. *Categories for Software Engineering*. Springer-Verlag, London, 2004.

[9] M. G. Hinchey and S. A. Jarvis. *Concurrent Systems: Formal Development in CSP*. International Series in Software Engineering. McGraw-Hill International, London, UK, 1995.

[10] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall International Series in Computer Science. Prentice Hall International, Englewood Cliffs, NJ, 1985.

[11] W. M. L. Holcombe. X-Machines as a basis for system specification. *Software Engineering*, 3(2):69–76, 1988.

[12] C. A. Rouff, W. F. Truszkowski, M. G. Hinchey, and J. L. Rash. Formal approaches to intelligent swarms. In *Proc. SEW-28, 28th Annual NASA/IEEE Software Engineering Workshop*, Greenbelt, MD, 2003. NASA Goddard Space Flight Center, IEEE Computer Society Press, Los Alamitos, Calif.

[13] C. A. Rouff, W. F. Truszkowski, M. G. Hinchey, and J. L. Rash. Verification of emergent behaviors in swarm based systems. In *Proc. 11th IEEE International Conference on Engineering Computer-Based Systems (ECBS), Workshop on Engineering Autonomic Systems (EASe)*, pages 443–448, Brno, Czech Republic, May 2004. IEEE Computer Society Press, Los Alamitos, Calif.

[14] W. M. Spears and D. F. Gordon. Using artificial physics to control agents. In *Proc. IEEE International Conference on Information, Intelligence, and Systems*, Charlotte, North Carolina, November 1999.

[15] R. Sterritt and M. G. Hinchey. Apoptosis and self-destruct: A contribution to autonomic agents? In *Proc. FAABS-III, 3rd NASA/IEEE Workshop on Formal Approaches to Agent-Based Systems*, pages 269–278. Springer-Verlag, April 2004.

[16] C. Tofts. Describing social insect behavior using process algebra. *Transactions on Social Computing Simulation*, pages 227–283, 1991.

[17] W. Truszkowski, M. Hinchey, J. Rash, and C. Rouff. NASA's swarm missions: The challenge of building autonomous software. *IEEE IT Professional*, 6(5):47–52, September/October 2004.

[18] W. F. Truszkowski, M. G. Hinchey, J. L. Rash, and C. A. Rouff. Autonomous and autonomic systems: A paradigm for future space exploration missions. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 2006 (to appear).